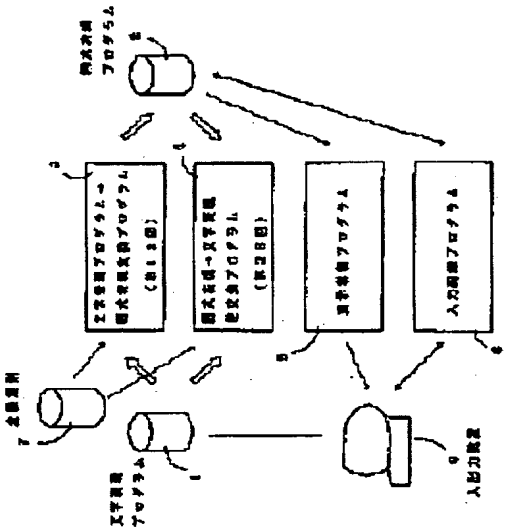
METHOD AND DEVICE FOR DISPLAY/GENERATION OF PROGRAM

Patent number: JP4127235
Publication date: 1992-04-28
Inventor: OSHIMA YOSHIMITSU
Applicant: HITACHI LTD
Classification:
- international: G06F9/06; G06F3/14; G09G5/00; G09G5/14; G09G5/40
- european:
Application number: JP19900281269 19901019
Priority number(s):

Abstract of **JP4127235**

PURPOSE:To display a program in an easy-to-understand graphic expression by analyzing the program expressed in characters for extraction of each group of partial structures of the program, assigning a graphic having an area to each partial structure, and putting a graphic on another in an area to show the enclosing relation between the partial structures.

CONSTITUTION:A given LISP PROGRAM is analyzed and a subject of conver sion is taken out and decided. If a list is decides as the subject of conversion, the heat key word of the list is extracted. Then, it is checked whether the key word is included in a conversion rule 7 or not. If so, the key word is converted into a specified graphic expression based on the contents of the rule 7. If not, the key word is converted into the graphic expression of a general function based on the contents of the rule 7. Thus, the partial structures of a program having an extent can be satisfactorily displayed in the graphic expressions.



⑲日本国特許庁（JP）　⑪特許出願公開

⑫公開特許公報（A）　平4－127235

⑤1Int. Cl.[5]		識別記号		庁内整理番号
G 06 F	9/06	4 3 0	G	7927－5B
	3/14	3 1 0	E	9188－5B
G 09 G	5/00		A	8121－5G
	5/14			8121－5G
	5/40			8121－5G

⑬公開　平成4年(1992)4月28日

審査請求　未請求　請求項の数 12　（全21頁）

⑤4発明の名称　プログラム表示方法および装置ならびにプログラム生成方法および装置

⑳特　願　平2－281269
㉒出　願　平2(1990)10月19日

優先権主張　㉜平2(1990)3月9日㉝日本(JP)㉛特願　平2－56360

⑦2発明者　大島　義光　東京都国分寺市東恋ケ窪1丁目280番地　株式会社日立製作所中央研究所内

⑦1出願人　株式会社日立製作所　東京都千代田区神田駿河台4丁目6番地

⑦4代理人　弁理士　有近　紳志郎

明　細　書

1．発明の名称

プログラム表示方法および装置ならびに
プログラム生成方法および装置

2．特許請求の範囲

1．文字表現によるプログラムを解析し、各々が一まとまりのプログラムの部分構造を抽出し、各部分構造に各々が領域をもつ図形を割り当て、その領域によりその内部で使用する変数の有効範囲およびプログラムとデータの区別を表現し、且つ、一つの図形の領域内に他の図形を重ねることによりそれら図形に対応する部分構造間の包含関係を表現することを特徴とするプログラム表示方法。

2．一まとまりのプログラムの部分構造が、主プログラム，副プログラム，局所的な変数定義と複数の実行文からなる複合文等のブロック構造および繰り返し等の制御の範囲およびその内部で使用する変数の有効範囲を示す命令列，または引用命令等のデータ指定記述により指定された一まとまりのデータ列であることを特徴とする請求項1のプログラム表示方法。

3．一まとまりのプログラムの部分構造の内容に応じて、対応する図形の形状を異ならせることを特徴とする請求項1または請求項2のプログラム表示方法。

4．一まとまりのプログラムの部分構造の内容に応じて、対応する図形に付与するキーワードを異ならせることを特徴とする請求項1から請求項3までのいずれかのプログラム表示方法。

5．一つの部分構造が他の部分構造を参照する場合において、他の部分構造の参照を示す図形を表示することと，その図形に置換して他の部分構造に対応する図形を表示することとを、任意に選択可能としたことを特徴とする請求項1から4までのいずれかに記載のプログラム表示方法。

6．参照が、副プログラムの呼び出し，マクロ

の参照，変数参照である請求項1から5までのいずれかに記載のプログラム表示方法。

7．文字表現によるプログラムを解析して各々が一まとまりのプログラムの部分構造と各部分構造の包含関係を抽出するプログラム解析手段と、

各部分構造に対して領域をもつ図形を割り当ててその領域によりその内部で使用する変数の有効範囲およびプログラムとデータの区別を表現すると共に、各部分構造間の包含関係により各部分構造に割り当てた図形を重ねる図式表現化手段と、

前記図式表現化された情報を画像で表示する画像表示手段と

を具備してなることを特徴とするプログラム表示装置。

8．各々が領域をもつ複数の図形が重なり合い且つ各図形に矢印が付された図式表現を解析して、各図形から一まとまりのプログラムの部分構造を抽出し、各図形の重なりの状態から前記部分構造の包含関係を抽出し、各図形の領域から前記部分構造の内部で使用する変数の有効範囲およびプログラムとデータの区別を抽出し、それら各プログラムの部分構造と包含関係とから文字表現によるプログラムを生成することを特徴とするプログラム生成方法。

9．各図形の形状的特徴から、その形状的特徴に対応する機能をもつ一まとまりのプログラムの部分構造を抽出することを特徴とする請求項8のプログラム生成方法。

10．各図形に付されたキーワードから、そのキーワードに対応する機能をもつ一まとまりのプログラムの部分構造を抽出することを特徴とする請求項8または請求項9のプログラム生成方法。

11．文字表現によるプログラムが、機械語命令列で記述されたプログラムであることを特徴とする請求項8から請求項10までのいずれかのプログラム生成方法。

12．各々が領域をもつ複数の図形が重なり合い且つ各図形に矢印が付された図式表現を解析して、各図形から一まとまりのプログラムの部分構造を抽出し、各図形の重なりの状態から前記部分構造の包含関係を抽出し、各図形の領域から前記部分構造の内部で使用する変数の有効範囲およびプログラムとデータの区別を抽出する図式表現解析手段と、

それら各プログラムの部分構造と包含関係とから，所定の順序で処理を記述した文字表現を生成する文字表現化手段と

を具備してなることを特徴とするプログラム生成装置。

3．発明の詳細な説明

【産業上の利用分野】

本発明は、プログラム表示方法および装置ならびにプログラム生成方法および装置に関し、さらに詳しくは、文字表現されたプログラムの処理内容を図式表現で表示する方法および装置ならびに図式表現された処理内容から文字表現によるプログラムを生成する方法および装置に関する。

【従来の技術】

最近、プログラムを図式表現により表示し、視覚的にその理解を容易にしようという提案が活発に行われている。

古くは、プログラムの図式表現の方法として、フローチャートがもっぱら用いられていた。しかし、フローチャートは処理の流れを表現することは出来るが、プログラムの階層的な制御構造をうまく表現できないという問題があつた。

近年、この問題を改良するものとして、いくつかの新しい図式表現によるプログラム表示方法が提案されている。また、これらのプログラム表示方法を用いて、プログラムを図式表現で表示し、プログラムの理解ないし作成を支援するシステムが実現されている。

これらの図式表現によるプログラム表示方法の公知例のいくつかが、文献「プログラム制御構造の新しい表現法：木構造チャートが普及期に：日経コンピュータ、1989.1.9、pp.71～83 」に紹介

されている。

そこに掲げられている図式表現によるプログラム表示方法の例を第３図(１)(２)(３)により説明する。

第３図(１)は、繰り返し，条件分岐，逐次実行などからなるFORTRANプログラムである。

第３図(２)は、同図(１)のプログラムの制御構造を、木構造状に配置した図式表現により表示したもので、ＰＡＤと呼ばれている。

この表示方法では、単位となる処理を箱に入れて表し、それを線で組み合わせてプログラムの制御構造を表現する。特定の処理の箱には、それを明示する形を付与している。例えば、「繰り返し」の処理の箱の形は、図の３１に示すように、箱の右端に縦棒を入れた形である。箱中の文字列は、Ｉを１から５まで変化させて、その右に接続されている箱の中の処理（Ａ（Ｉ）＝Ｂ（Ｉ）＋Ｃ（Ｉ））を繰り返し実行することを表している。また、「条件分岐」の処理の形は、図の３２に示すように、箱の右端の形をくの字形にした形である。箱中の文字列は、条件「Ｘ＞Ｙ」が成立したときは、箱３２の上辺に接続されている箱の処理（Ｚ＝Ｘ）を実行し、条件が成立しないときは、箱３２の下辺に接続されている箱の処理（Ｚ＝Ｙ）を実行することを表している。また、箱３１と箱３２の左端を縦棒で結ぶことによって、箱３１の処理と箱３２の処理を上から順に実行する「逐次実行」の処理を表している。なお、「繰り返し」，「条件分岐」，「逐次実行」以外のプログラムの制御構造は、箱中に文字表現で表示する。

同図(３)は、同図(１)のプログラムの制御構造を、入れ子構造状の図式表現により表示したもので、ＮＳチャートと呼ばれている。

この表示方法も、「繰り返し」，「条件分岐」，「逐次実行」の３つの制御構造以外の制御構造は、箱中に文字表現で表示する。

これらの図式表現によるプログラム表示方法は、基本的に構造化プログラミングの考え方に則ったもので、FORTRAN，ＰＬ／Ｉ，PASCAL，Ｃなどの手続き型プログラミング言語におけるプログラムの制御構造を図式化することを主眼としている。

ところで、ＰＬ／Ｉ，PASCAL，Ａｄａ，Ｃおよび Common LISP などの近代的なプログラミング言語では、プログラムを構造的に記述する言語要素として、上記３つの制御構造以外に、内部サブルーチンや内部関数あるいはプログラム中で局所的な変数を導入するいわゆるブロック構造と呼ばれている制御構造を用意している。

第４図に、ブロック構造を用いて記述したＰＬ／Ｉプログラムの一例を示す。同図の行番号２から５までが内部関数の記述、行番号８から１１までが局所的変数を導入した複合文の記述である。行番号６と９で同名の変数Ｃを定義（宣言）しているが、両者は別の変数を表している。また、行番号９で定義（宣言）している変数Ｃの有効範囲は、行番号８から１１までであり、その外側では行番号６で定義（宣言）している変数Ｃが有効となる。したがって、行番号１２の出力文では、行番号７で与えられた計算結果８が出力される。

このような内部関数または局所変数定義付き複合文の記述を複数用いて、多重の入れ子構造とすることも出来る。従って、モジュール性の良い階層化されたプログラムを記述することが出来る。

上記ＬＩＳＰなどの関数型言語で記述されたプログラムを図式表現により表示する方法の例を、第５図(１)(２)(３)により説明する。

第５図(１)は、ＬＩＳＰプログラムである。

同図(２)は、同図(１)のプログラムの関数の呼び出し関係を木構造状に表わしたものである。言語要素である関数を箱で表し、箱の中に関数名を記述する。また、それぞれの箱の間を矢印で結ぶことによって各関数間の入出力関係を表している。この例は、文献「Levien.R.:Visual Programming : Byte, February, 1986, pp. 135～144」などに紹介されている方法である。

同図(３)は、同じプログラムを、入れ子構造状の図式表現により表わしたものである。各閉図形が一つ一つの関数に対応し、その形が関数の機能を表わしている。この例は、文献「Chang, S-K., Ichikawa, T. and Ligomenides, P. A. :Visual

Languages : Plenum Press, 1986, p. 52」に紹介されている方法である。

ところで、ＬＩＳＰでは、第５図(１)に示したように、カッコを用いたリストと呼ばれる形式でプログラムを記述する（リストとは要素の並びをカッコで囲ったもの。要素がまたリストであっても良い）。しかし、ときによって、リストをプログラムとしてではなく、データとして扱いたい場合がある。ＬＩＳＰでは、データとしてのリストをプログラムと区別するために、quote という特殊関数を用意している。この quote を先頭に置いたリストは、それ以降の部分をプログラムとして実行せず、データとして取り扱うことを意味する。なお、ＬＩＳＰ言語における用語では、プログラムを実行することを、プログラムを「評価する」という。また、特殊関数 quote を先頭に付けてリストをデータとして扱うことを、「引用する」という。したがって、特殊関数 quote は、引用命令に相当する。

第６図(１)に特殊関数 quote を用いたプログラム例を示す。第６図(２)は、第６図(１)のプログラムを前記第５図(２)の方法で図式表現したものである。

【発明が解決しようとする課題】

前記ＰＡＤやＮＳチャートのようなプログラム表示方法は、ブロック構造の図式表現を持っていない。したがって、第４図のようなブロック構造をもつ階層化されたプログラムを図式表現でうまく表示することが出来ない問題点がある。

また、「繰り返し」，「条件分岐」，「逐次実行」の３つの制御構造の図式表現についても、問題点がある。

例えば、前記ＰＡＤの繰り返しの表現に、以下のような問題点がある。

すなわち、ＰＡＤでは、前記第３図(２)で示したように、繰り返しの制御部分を箱中に文字表現で記述している。しかし、C や Common LISP などでは、繰り返しの制御部分に任意のものを書くことが出来るため、この部分が複雑な内容になる場合がある。この場合に、その複雑な内容を繰り返しの箱の中に全て文字表現で入れると、表示が煩雑で見にくいものになる。

また、繰り返しの制御は通常制御変数を用いて行うが、この制御変数には前記ブロック構造と同様に有効範囲の概念がある。しかし、ＰＡＤによる図式表現形式では、この制御変数の有効範囲を明確に表わすことが出来ない。

一方、第５図(２)の従来方法でも、第６図(２)から分かるように、データとして扱うべき部分（第６図(１)の（plus　５　６）の部分）も関数と同じ図式表現で表わされるため、データとしてのリストを明確にプログラムと区別して見ることが出来ない問題点がある。

このように、従来の図式表現によるプログラム表示方法では、広がりを持ったプログラムの部分構造（ブロック構造，繰り返し，データとしてのリスト）をうまく表示できない問題点がある。

そこで、本発明の目的は、広がりを持ったプログラムの部分構造をも含めて、プログラムを図式表現で分かりやすく表示する方法および装置を提供し、これによってプログラムを視覚的に容易に理解出来るようにすることにある。

また、本発明の他の目的は、本発明に係る図式表現を用いて一連の処理を入力・編集したものから、文字表現によるプログラム（ソースプログラムまたは機械語命令列）を生成する方法および装置を提供し、これによってプログラムの作成を容易にすることにある。

【課題を解決するための手段】

本発明は、第１の観点では、文字表現によるプログラムを解析し、各々が一まとまりのプログラムの部分構造を抽出し、各部分構造に各々が領域をもつ図形を割り当て、その領域によりその内部で使用する変数の有効範囲およびプログラムとデータの区別を表現し、且つ、一つの図形の領域内に他の図形を重ねることによりそれら図形に対応する部分構造間の包含関係を表現することを特徴とするプログラム表示方法を提供する。また、そのプログラム表示方法を実施する装置を提供する。

上記構成において、一つの部分構造が他の部分

構造を参照する場合に、他の部分構造の参照を示す図形を表示することと，その図形に置換して他の部分構造に対応する図形を表示することとを、任意に選択可能とするのが好ましい。

本発明は、第２の観点では、各々が領域をもつ複数の図形が重なり合い且つ各図形に矢印が付された図式表現を解析して、各図形から一まとまりのプログラムの部分構造を抽出し、各図形の重なりの状態から前記部分構造の包含関係を抽出し、各図形の領域から前記部分構造の内部で使用する変数の有効範囲およびプログラムとデータの区別を抽出し、それら各プログラムの部分構造と包含関係とから文字表現によるプログラムを生成することを特徴とするプログラム生成方法を提供する。また、そのプログラム生成方法を実施する装置を提供する。

【作用】

前記第１の観点によるプログラム表示方法および装置では、まず、文字表現によるプログラムを解析し、そのプログラムから、主プログラム，副プログラム，局所的な変数定義と複数の実行文からなる複合文等のブロック構造および繰り返し等の制御の範囲およびその内部で使用する変数の有効範囲の概念を有するプログラムの部分構造を与える一まとまりの命令列（一つの命令であってもよい）を抽出する。また、引用命令等のデータ指定記述により指定された一まとまりのデータ列（一つのデータであってもよい）を抽出する。また、それら命令列およびデータ列の包含関係を抽出する。次に、前記命令列およびデータ列を代表する命令または命令列を、閉図形などの領域をもつ図形に変換する。また、その他の要素を、前記図形の内部に含まれる図形または文字表現に変換する。次に、前記包含関係に従って前記図形を重ね合わせる。以上により、文字表現で与えられたプログラムを図式表現で表示できる。。

この図式表現によれば、領域を持つ図形とその重なりとにより、広がりを持ったプログラムの部分構造（ブロック構造，繰り返し，データとしてのリスト）をうまく表示できるようになる。

前記第２の観点によるプログラム生成方法および装置では、図式表現を解析し、閉図形などのまとまりの範囲を示す図形と，その内部に含まれる図形または文字表現と，前記図形の重なりとから、主プログラム，副プログラム，局所的な変数定義と複数の実行文からなる複合文等のブロック構造および繰り返し等の制御の範囲およびその内部で使用する変数の有効範囲の概念を有するプログラムの部分構造を抽出する。また、それらプログラムの部分構造の包含関係を抽出する。次に、前記プログラムの部分構造の機能と包含関係とにより、命令列またはデータ列を生成する。以上により、図式表現から、それに対応する文字表現のプログラムが得られる。

【実施例】

以下、実施例を用いて本発明をさらに詳細に説明する。なお、これにより本発明が限定されるものではない。

第７図に、本発明のプログラム表示方法を実現するプログラム表示装置を示す。

１１は、２次元または３次元の図形表示能力を有するデイスプレイで、この画面に図式表現でプログラムを表示する。

１２は、文字を入力したり，装置に対する各種の指示を入力するキーボードである。

１３は、デイスプレイ１１の画面上の位置の指示等を行うためのマウスなどのポインテイング装置である。

１４は、パーソナルコンピュータやワークステーション等における計算機装置であり、中央演算処理装置いわゆるＣＰＵや，主記憶装置や，前記デイスプレイ１１に図形や文字を表示するためのディスプレイコントローラなどを内蔵している。また、必要に応じて、ハードデイスクやフロッピーデイスクなどの２次記憶装置を内蔵する。主記憶装置は、本発明のプログラム表示方法を実行するための各種プログラムが記憶されている。また、図式表現で表示する対象となる文字表現によるプログラムおよびデータが記憶されている。

第２図(１)は、ＬＩＳＰ言語による階乗関数

（factorial）のプログラムである。このプログラムの動作は次のとおりである。

引数ｎ（第２図(１)の１行目）に具体的な数値が与えられてこのプログラムが呼ばれると、まず局所変数 result を〈１〉に初期化する(２行目)。次にｉを１からｎまで変化させて（３，４行目）、順にｉを繰り返し result に掛けていき，その結果をまた result に置き換える（５行目）。ｉがｎより大きくなったところでdoループを終了し，result を返す（４行目）。要するに、１からｎまでを掛けあわせた結果を返す。

第２図(２)は、本発明のプログラム表示方法により第２図(１)のＬＩＳＰプログラムを図式表現で表示した例である。

２０は、関数 factorial の定義を示す枠である。枠２０の上辺に、左から順に、関数定義であることを示すキーワード defun、定義する関数名 factorial、引数ｎをそれぞれ箱に入れて示している。

枠２０の内部に関数定義の本体がある。すなわち、局所変数定義letの枠２１で囲まれた部分が関数定義本体となっている。

枠２１も、枠２０と同様に、上辺に、左から順に、局所変数定義であることを示すキーワード let、そこで定義している局所変数 result をそれぞれ箱に入れて示している。

局所変数 result の箱に右方から入っている矢印２１１および矢印の右端の数値〈１〉によって、局所変数定義 let の枠２１に制御が移ったとき、局所変数 result に初期値〈１〉が与えられること示している。

枠２１の中には、繰り返しの制御構造 do を示す枠２２がある。枠２０，２１と同様に、上辺に、左から順に、キーワード do、繰り返しの制御変数ｉを入れた箱がある。

繰り返しの制御変数は、通常初期値と２回目以降の更新値を指定する必要があるが、それは制御変数ｉの箱の右側に並べたキーワード init および step で示されている。キーワード init に左方から入る矢印によって与えられている数値〈1〉が実際の初期値である。２回目以降の更新値を示すキーワード step に左方から入る矢印につながっている箱２２１から実際の２回目以降の更新値が与えられる。

箱２２１は、関数を表す図式表現である。一般に、関数を表わす箱の右方から入るパラメタ（複数でもよい）の値をもとに、箱の中に示したキーワードに対応する演算を行い、左方から出ている矢印を経由して、矢印が指している要素に演算結果の値を渡す。箱２２１の場合は、パラメタｉに、１を加え（キーワード（1+）に対応する演算）、その結果を制御変数ｉに与え、制御変数を更新する。箱２２１およびパラメタは、第２図(１)の３行目の関数表現（1+ i）に対応している。

枠２２の中にある２つの三角形の頂点を対向させた図形２２２は、繰り返しの制御構造 do の終了判定部分を表わしている。

図形２２２の上方から入っている矢印２２３は、終了判定の条件を表わし、この矢印によって示される値が nil（nilはＬＩＳＰ言語の世界で論理条件の「偽」を表わす記号）以外のときは、終了条件が成立し、図形２２２に右方から入っている矢印２２４につながる部分（局所変数 result ）が評価されて、繰り返しの制御構造 do の値として左方から出ている矢印を経由して出力される。矢印２２３によって示される値が nil のときは、終了条件が成立せず、図形２２２の下方に伸びる線分２２６につながる処理すなわち繰り返しの処理本体が実行される。

箱２２５は、終了条件を示す関数である。この例では、変数ｉと変数ｎとの大小比較を行って、ｉ＞ｎのときは値ｔ（ＬＩＳＰ言語の世界で論理条件の「真」を表わす記号）を出力し、ｉ＞ｎでないときは nil を出力する。

箱２２７，箱２２８は、繰り返しの処理本体である。この例では、変数ｉと変数 result を掛け合わせ（箱２２７で示す関数「＊」）、結果を変数 result に代入する（箱２２８で示すＬＩＳＰの特殊関数「setq」）。

以上のように、本発明の図式表現によるプログ

ラム表示方法および装置によれば、第２図(１)のＬＩＳＰプログラムが、第２図(２)のように図式表現される。

これにより、プログラムのブロック構造および繰り返しのまとまりの範囲が良く分るようになる。また、関数定義の引数，局所変数，繰り返しの制御構造の制御変数の有効範囲も良く分るようになる。

第１１図(ａ)(ｂ)の表に、ＬＩＳＰで用いられる個別のプログラム言語要素と図式表現との対応関係を示す。これらの個々については後で詳述する。

第８図(１)は、プログラミング言語Ｃによる階乗関数のプログラムである。第８図(２)は、第８図(１)のプログラムを、本発明のプログラム表示方法により表示した例である。

第９図(１)は、プログラミング言語ＰＬ／Ｉによる階乗関数のプログラムである。第９図(２)は、第９図(１)のプログラムを、本発明のプログラム表示方法により表示した例である。

第８図(２)および第９図(２)の例においても、第２図(２)の例と同様、関数定義，局所変数定義のブロック構造，繰り返しの制御構造が枠（第８図(２)の４１，４２，４３および第９図(２)の５１，５２，５３）によって表わされており、ブロック構造および繰り返しの制御のまとまりの範囲が良く分かる。また、関数定義の引数，局所変数，繰り返しの制御変数の有効範囲が良く分る。

第１０図は、第６図(１)のＬＩＳＰプログラムを、本発明のプログラム表示方法により表示した図式表現である。

データとしてのリストを記述するための特殊関数 quote を、二重線の枠６０で図式表現することにより、第６図(２)の従来の図式表現では判然としなかったプログラム部分とデータ部分の区別が明確になっている。

以下、本発明によるプログラム表示方法および装置の具体的内容について説明する。なお、本実施例では、本発明によるプログラムの表示および生成方式を含むプログラムの図式化処理の全体について述べる。

第１図は、本発明のプログラム表示方法の処理の概略を示すブロック図である。

１は、第２図(１)にあるような文字表現のプログラムである。

２は、第２図(２)で示したような図式表現により表わされたプログラムである。

７は、文字表現によるプログラムを、対応する図式表現に変換する変換規則である。文字表現によるプログラムに含まれる主プログラム，副プログラム，局所的な変数定義と複数の実行文からなる複合文等のブロック構造，繰り返し等の制御の範囲，繰り返し等の制御の内部で使用する変数の有効範囲等のプログラムの部分構造を与える命令列および引用命令等のデータ指定記述により指定された一まとまりのデータ列を、対応する図式表現に変換する変換規則が含まれる。例えばＬＩＳＰ言語の場合、第１１図(ａ)(ｂ)に示すものである。

３は、前記変換規則７を参照して、文字表現によるプログラム１を、図式表現に変換する変換プログラムである。

４は、前記変換規則７を参照して、図式表現により表わされたプログラム２から、文字表現によるプログラム１を生成する逆変換プログラムである。

５は、図式表現により表わされたプログラム２を取り出して入出力装置８に送り出し画面表示したり、その表示方法を制御するための表示制御プログラムである。

６は、入出力装置８を通して、図式表現により表わされたプログラム２に修正を加えたり、図式表現により表わされたプログラム２を新たに入力したりする入力編集プログラムである。

第１２図に、変換プログラム３の基本処理の流れを示す。本図では、処理の流れを示すためにＰＡＤによる表現を使用している。また、文字表現によるプログラム１として、ＬＩＳＰプログラムを想定している。

ステップ１０１では、与えられたＬＩＳＰプロ

グラムを解析し、変換対象（プログラムの部分構造を与える命令列または一まとまりのデータ列）を取りだし、それがリストであるか，シンボルまたはリスト以外のデータであるかを判定する。リストとは、前述したように、開きカツコと閉じカツコで囲まれた複数の要素の並びである。シンボルとは、通常１つ以上の文字の並びからなる文字列で、変数の名称などを表わすために用いられる。リスト以外のデータとは、数値や，データとしての文字や文字列や，ベクトルや配列などであり、Common LISP 言語ではそれぞれ専用の表現法が与えられている。

ステップ１０１で変換対象がリストであると判定されると、ステップ１０１１に進む。変換対象がシンボルまたはリスト以外のデータであると判定されると、ステップ１０１３に進む。

ステップ１０１１では、リストの先頭のキーワードを抽出する（通常、リストの先頭の要素は、関数等の名称を表すキーワードである）。

ステップ１０１２では、そのキーワードが前記変換規則７にあるかどうかを調べる。

そのキーワードがあれば、特殊な図式表現への変換を必要とするので、ステップ１０１２１に進み、変換規則７の内容にしたがって特殊な図式表現への変換を行う。この処理は第１４図～第２１図を参照して後で詳述する。

そのキーワードがなければ、ステップ１０１２２に進み、変換規則７の内容にしたがつて一般関数の図式表現への変換を行う。この処理は第１３図を参照して後で詳述する。

一般関数としての図式表現の例は、第１１図(a)の変換規則表の項番１に示したものである。特殊な図式表現の例は、第１１図(ａ)の変換規則表の項番２～１０に示したものである。

ステップ１０１３では、変換対象（シンボルまたはリスト以外のデータ）をそのまま（文字のまま）出力する。この例を第１１図(ｂ)の変換規則表の項番１１～１２に示す。

最後に、ステップ１０２で、出力を表わす矢印を付ける。

次に、第１３図を参照して前記ステップ１０１２２の処理を詳細に説明する。

まず、ステップ２０１およびステップ２０１１で、関数の各引数部分について図式表現への変換を行う。各引数についての処理すなわちステップ２０１１の処理は、第１２図の基本処理と同じ内容となるため、第１２図の基本処理を再帰的に呼び出して行う。

次に、ステップ２０２で、関数を示す図式表現である箱を描き、その中に関数名を書き込む。

そして、ステップ２０３で、各引数の図式表現の出力と、関数の箱とを矢印で結ぶ。

次に、第１４図～第２１図を参照して前記第１２図のステップ１０１２１の処理を詳細に説明する。

第１４図は、局所変数定義のための関数 let の図式への変換の処理（１０１２１ｂ）を示している。

まず、ステップ３０１で、名前 let を表示する。

次に、ステップ３０２およびステップ３０２１，ステップ３０２２で、局所変数定義部分の処理を行う。ステップ３０２１では、定義する各局所変数ごとに、その初期値部分の図式表現への変換を行う。この処理は、第１２図の基本処理を再帰的に呼び出して行う。ステップ３０２２では、局所変数定義を示す箱を描き、変数名をその箱の中に書く。

ステップ３０３は、関数let の本体部分の処理である。関数let の本体部分は、Common LISP の言語仕様によれば、逐次実行 progn の処理と同様で、順に並べられた処理を逐次的に実行する処理である。したがつて、図式表現への変換も逐次実行 progn と同じ内容となるため、第１５図の逐次実行 progn の変換処理を呼び出して行う。

最後に、ステップ３０４で全体を枠で囲む。これにより、第１１図(ａ)の変換規則表のNo.３の図式表現となる。

第１５図は、逐次実行 prognの変換処理（ステップ１０１２１ｅ）を示している。

ステップ４０１およびステップ４０１１で、逐次実行 progn の中に並べられた各処理ごとに第１２図の基本処理を呼び出し、図式表現への変換を行う。

その後、ステップ４０２で、各処理を変換して得られた図式表現の左側（出力側）を縦線で結ぶ。これにより、第１１図(ｂ)の変換規則表のNo.６の図式表現となる。

第１６図は、関数定義のための特殊関数 defun の変換処理（ステップ１０１２１ａ）を示している。

まず、ステップ４１１で、名前 defun を表示する。

次に、ステップ４１２で、定義する関数名を表示する。

さらに、ステップ４１３およびステップ４１３１で、引数部分の処理を行う。

ステップ４１４では、前記第１４図のステップ３０３と同様に、第１５図の逐次実行 progn の変換処理を呼び出して本体部分の変換を行う。

最後に、ステップ４１５で全体を枠で囲む。これにより、第１１図(ａ)の変換規則表のNo.２の図式表現となる。

第１７図は、繰り返し do の変換処理（ステップ１０１２１ｃ）を示したものである。

まず、ステップ４２１で、名前 do を表示する。

次に、ステップ４２２およびステップ４２２１，ステップ４２２２、ステップ４２２３で繰り返しの制御変数部分の処理を行う。初期値部分，ステップ値部分それぞれについて第１２図の基本処理を呼び出して変換を行い、対応する制御変数の箱を描き、変数名をその中に書くことを繰り返す。

次に、ステップ４２３，４２４，４２５で、繰り返しの終了判定処理の部分の図式表現への変換を行う。引数の〈終了テスト〉および〈結果〉各々を第１２図の基本処理を呼び出して図式表現へ変換する。最後に、終了判定を表わす図形（２つの三角形を対向させて、その頂点を合わせた図形）を表示する。

次に、ステップ４２６で、第１５図の逐次実行 progn の変換処理を呼び出して本体部分の図式表現への変換を行う。

最後に、ステップ４２７で全体を枠で囲む。これにより、第１１図(ａ)の変換規則表のNo.４の図式表現となる。

第１８図は、関数 mapcar の変換処理（ステップ１０１２１ｄ）を示したものである。

ＬＩＳＰは一般に map 関数と呼ばれる一群の関数を持っているが、これは、リストを入力として、指定された処理を施し、結果をまたリストとして返す関数群である。いわゆる集合に対する写像（mapping）に類似した機能を果たすので、この名が付いている。関数 mapcar もその一つで、リストを入力として、その個々の要素に指定された処理を施し、個々の要素に対する処理結果を要素とするリストを返す。

まず、ステップ５０１で、名前 mapcar を表示する。

次に、ステップ５０２で、引数の〈リスト〉部分の変換処理を行う。この変換処理は、第１２図の基本処理を再帰的に呼び出して行う。

さらに、ステップ５０３で、引数の〈関数〉部分の変換処理を行う。この変換処理も、第１２図の基本処理を再帰的に呼び出して行う。

そして、ステップ５０４で、データの流れを示す図形すなわち〈関数〉の前後の三角形を表示する。

最後に、ステップ５０５で全体を枠で囲む。これにより、第１１図(ａ)の変換規則表のNo.５の図式表現となる。

第１９図は、条件分岐 if の変換処理を示したものである。

まず、ステップ５１１で、第１２図の基本処理を再帰的に呼出し、〈条件〉部分の図式表現への変換を行う。

次に、ステップ５１２で、第１２図の基本処理を再帰的に呼出し、〈処理１〉部分（〈条件〉が「真」であるときの処理）の図式表現への変換を行う。

さらに、ステップ５１３で、第１２図の基本処

理を再帰的に呼出し、〈処理２〉部分（〈条件〉が「偽」であるときの処理）の図式表現への変換を行う。

最後に、ステップ５１４で、条件分岐を表わす図形を表示する。これにより、第１１図(ｂ)の変換規則表のNo.７の図式表現となる。

第２０図は、代入 setq の変換処理を示したものである。

まず、ステップ５２１で、第１２図の基本処理を再帰的に呼出し、〈処理〉部分の図式表現への変換を行う。

次に、ステップ５２２で、〈変数〉の箱を描き、その中に変数名を入れる。

最後に、ステップ５２３で、全体を囲む箱を描く。これにより、第１１図(ｂ)の変換規則表のNo.８の図式表現となる。

第２１図は、特殊関数 quote の変換処理を示したものである。

関数 quote の引数の〈対象〉で与えられるリストまたはシンボルなどは、取り合えずデータとして取り扱われるとしても、その後の処理によつては、〈対象〉をやはりプログラムとしての関数または変数として取り扱いたい場合がある。本発明に係る図式表現では、プログラムとして表現する場合とデータとして表現する場合とでは図式表現を異ならせるので、データとして取り扱うのかプログラムとして取り扱うのかを判断する必要がある。

この判断は、前後の文脈により行える場合がある。例えば前記 map 関数の引数の〈関数〉として与えられた場合である。しかし、一般にはユーザが指定してやる必要がある。そこで、例えば、デフォルトではデータ形式で表現するとしておき、必要に応じて個別にデータ表現またはプログラム表現をユーザが指定できるようしておけばよい。

以上の点を考慮した関数 quote の変換処理をを第２１図に示す。

まず、ステツプ６０１で、〈対象〉をプログラムで表現すべきかデータで表現すべきかを判定する。

プログラムで表現すべきと判定した場合、ステツプ６０１１に進み、第１２図の基本処理を呼び出して、〈対象〉を図式表現に変換する。

データとして表現すべきと判定した場合、ステツプ６０１２に進み、対応する図式表現へ変換する。ここで、〈対象〉がリストの場合は、木構造状の図式表現とする。リスト以外の場合は、そのまま（文字のまま）出力する。

最後に、ステツプ６０２では、全体を二重線による枠で囲む。これにより、〈対象〉をプログラムとして表現するように指定されている場合、第６図(１)のプログラムが与えられると、第１０図のような図式表現が得られる。また、〈対象〉をデータとして表現するように指定されている場合、第２２図（１）の入力が与えられると、第２２図(２)に示すような木構造状の図式表現となる。

次に、第１１図(ｂ)の変換規則表の項No.１０のバッククォートについて説明する。

バッククォート（backquote）は、関数 quote の機能を発展させたもので、対象の前にバッククォート（逆引用符）を付けて表わす。関数 quote と同様に、一応それによつて指定された対象をプログラムではなくデータとして扱うが、さらにそのなかの項目の前にコンマが付いている場合、再度評価の対象にする。従って、リストとして表現されたデータのなかに、プログラムで評価して得られた結果のデータを埋め込むことが出来る。

第１１図(ｂ)の変換規則表の項No.１０のバッククォートの図式表現は、上記の機能を表わすもので、評価しない部分には影をつけて表わし、評価する部分を白抜きにして表わしている。バッククォートは多重にして使用することもできるが、その場合は影を重ねて表現する（つまり、重なつた部分の影が濃くなる）。また、影でなく、ハツチングで表現してもよい。

以上のことを考慮したバッククォートの図式表現への変換処理を第２３図(１)(２)に示す。同図（１）がバッククォートそのものに対する処理、（２）がコンマの付いたリストに対する処理である。

（１）では、ステップ７０１で、バッククォートを除いた部分の図式表現への変換処理を行う。これは、第１２図の基本処理を再帰的に呼び出して実行する。

ステップ７０２で、全体を枠で囲み、白抜き指定部分を除き、枠の中の影付けを行う。

（２）では、ステップ７１１で、コンマを除いた部分の図式表現への変換処理を行う。これは第１２図の基本処理を再帰的に呼び出して実行する。

ステップ７１２で、内部が透明の枠をつける。この枠は、上述の（１）のステップ７０２での白抜き指定のためのものである。

以上、第１１図(ａ)(ｂ)に示したプログラム構成要素に対応する変換処理を中心に，変換プログラム３の処理の流れについて説明した。

なお、第１１図(ａ)(ｂ)に示していないプログラム構成要素もＬＩＳＰにはある。それらは第１１図(ａ)(ｂ)の図式表現と類似の図式表現を用いて表示可能である。例えば、無名関数の定義のための記述要素 lambda の図式表現は、関数定義 defun の図式表現から関数名称の部分を除いたものを用いればよい。また、非局所的脱出の範囲を定義する block，goto 文（ＬＩＳＰでは特殊関数 go で実現）のラベルの有効範囲を規定する tagbody，prog なども局所変数定義 let に類似の図式表現で表示可能である。さらに、Common ＬＩＳＰの特徴である列型のデータ（列型とは、一次元の配列であるベクトル型と一次元のリスト型とを併せた抽象的なデータ型）に対する処理関数の一部は、map 関数の図式表現を利用して表示することが出来る。

なお、以上の実施例では、プログラム構造などの一まとまりのプログラムの部分構造を示す図式表現として矩形の枠を用いたが、プログラムまたはデータのまとまりの範囲を示す任意の図形を用いてもよい。例えば、他の閉図形を用いてもよいし、また、第２４図に示すような線の太線によってブロック構造を図式表現してもよい。

また、以上の実施例では、プログラムを全て図式表現に変換することを前提としていたが、文字表現のプログラムを部分的に残してもよい。例えば、通常の文字表現のプログラムに本発明における枠による構造表現をかぶせることによつて、プログラムの理解度を上げることが出来る。これを実現するためには、上記実施例で示した変換処理を部分的に適用すればよい。

次に、逆変換プログラム４について説明する。

逆変換は、ディスプレイ装置またはプリンタに出力されたプログラムの図式表現をパターン認識技術によつて読み取り、それを解析して図式表現の構造データ（図式表現の各要素とそれらの間の関係）を作成し、その構造データからプログラムの部分構造を生成し、各部分構造を連結して文字表現によるプログラムを生成することにより実現できる。しかも、ディスプレイ装置またはプリンタに出力される基となった図式表現の構造データがあれば、それを利用できるので、図式表現をパターン認識技術によつて読み取る処理は不要になる。

構造データは、通常、文字表現によるプログラム１を図式表現により表示する際に、内部的に作成される。また、第１図の入力編集プログラム６を用いて図式表現を入力した場合にも、作成される。さらに、構造データは、第１図の入力編集プログラム６を用いて修正することも可能である（修正された場合は、修正結果に対応する図式表現プログラムが表示される）。

第２５図に、構造データの例（部分）を示す。これは第２図(１)に示した階乗関数の例である。

この構造データは、複数の要素（以下、これをセルとよぶ）を連結したリスト構造になっている。

セルは、プログラムの部分構造や部分構造で用いられるデータなどに対応しており、部分構造に対応する名称のスロットを持っている。また、図式表現の表示に必要な表示位置，表示サイズの情報を持つている。表示位置，表示サイズの情報は、文字表現によるプログラムから図式表現への変換時もしくは図式表現の入力／編集時に設定される。

さらに、セルは、プログラムの部分構造に引数や変数や本体などがある場合、引数リストや変数

リストや本体に対する付加的なスロットを持つ。引数リストや変数リストや本体そのものは別のセルとして設定され、前記付加的なスロットにつながれる。なお、第２５図には２つの箱からなる図形（１６１～１６４）があるが、これはスロットに対する下位のレベルのセルが複数個存在する場合に、それらをまとめて一つのリストとするために用意されている（ただし、第２５図の例では、複数項目設定可能なスロットに設定されている項目数も、全て１つとなつている）。

第２６図は、第２５図の構造データから文字表現のプログラムを生成する逆変換プログラム４を示している。

まず、ステツプ８０１で、構造データの最上位のセルを見て、引数リストや変数リストや本体などの付加的なスロットがあるかどうかを調べる。

あれば、ステツプ８０１１に進み、付加的スロットごとに、第２６図（本図）の処理を再帰的に呼び出し、それらの付加的なスロットの指すセルの内容に対応する文字表現を生成する（８０１１）。

次に、ステツプ８０２に進み、第１１図(a)(b)の変換規則表を参照し、セルの内容を文字表現によるプログラムに変換する。この変換処理は、第１３図～第２１図および第２３図で示した変換処理を逆に読み変えてやれば実現できる。

構造データの最上位のセルについてのステップ８０２の処理が終れば、文字表現によるプログラム全体が得られることになる。

得られた文字表現によるプログラムは、ＬＩＳＰなどのインタプリタ言語ではそのまま実行可能である。ただし、これを機械語に変換すればより高速の実行が可能となるので、図式表現を機械語のプログラムに変換するように変換規則を作成しておき、図式表現から機械語のプログラムを直接生成するようにしてもよい。

なお、図式表現から文字表現（高級言語または機械語）によるプログラムを生成せずに、図式表現のままで実行させることも、インタプリタ技術を用いることによつて、実現可能である。

次に、第１図の表示制御プログラム５について説明する。

図式表現を見ながらプログラムを理解していく過程で、そのプログラムに関連する情報、具体的には、そのプログラムの中で参照している副プログラム，関数，マクロの定義内容，変数の定義または初期値ないし現在値などを見たい場合がある。

このような場合、従来は、表示画面の一部にそれを出すとか、最近のマルチウインドウを採用しているシステムでは、別のウインドウを開き、そこに表示する、などの方法が採られている。

このような表示の場合、参照箇所とその参照内容は、ユーザの頭の中で関連づけられるだけで、不確かなものである。

これに対して、表示制御プログラム５は、ユーザの指示に従って、参照箇所に参照内容を埋め込んで表示し、関連を視覚的に明示する機能を有している。

第２７図(１)(２)(３)により、この機能を説明する。

第２７図(１)は、参照箇所に、他を参照することを表わす図形９０１を表示した状態である。

同図(２)は、その参照内容を表わす図形である。

同図(３)は、同図(１)の参照箇所の図形９０１を、同図(２)の図形に置き換えて表示した状態である。

ユーザは、第２７図(１)または同図(３)の表示を任意に選択できる。

第２７図(１)の表示の場合は、全体構造を把握しやすい。また、第２７図(３)の表示の場合は、参照箇所と参照内容が視覚的に一体のものとして表示され、データの入出力も含めた両者の関係が一目瞭然に分かるので、より詳細にプログラムを理解できる。

表示制御プログラム５は、第２５図の構造データのセルの変更により、第２７図(１)または同図(３)の表示の切り替えを行う。例えば、ユーザ関数の呼び出しの図式表現に対応するセルに関数定義のスロットを設けて、関数参照箇所にその定義内容を埋め込んで、第２７図(１)の表示を行う。

そして、ユーザの指示があると、前記関数定義のスロットにその定義内容のデータを入れて、第２７図(３)の表示を行う。

なお、参照内容の図形が大きくて、参照箇所に表示すると元の図式表現と大きく変わってしまう場合は、部分的に参照内容を表示したり，縮小して表示するなどの対応により、元の図式表現とあまり変わらないようにするのが好ましい。

以上の実施例ではＬＩＳＰプログラムを用いて説明したが、本発明は、FORTRAN，Ｃ，ＰＬ／Ｉ，PASCAL などの他のプログラミング言語にも適用可能である。すなわち、それぞれの言語の基本機能に応じて変換規則を適切に設定すれば、上記説明の処理を用いて、文字表現によるプログラムを図式表現で表示できる。また、逆に、図式表現から各言語のプログラムを生成することが出来る。

【発明の効果】

本発明のプログラム表示方法および装置によれば、広がりを持ったプログラムの部分構造（ブロック構造，繰り返し，データとしてのリスト）をうまく図式表現で表示できるようになる。このため、プログラムを視覚的に容易に理解できるようになる。また、参照内容を参照場箇所において表示するため、この点でもプログラムを視覚的に容易に理解できるようになる。

他方、本発明のプログラム生成方法および装置によれば、図式表現から、それに対応する文字表現のプログラムを得ることが出来るようになる。図式表現を用いれば、プログラマの意図を明確に表現しやすいから、プログラマの意図どおりのプログラムを容易に作成できるようになり、生産性が向上する。

４．図面の簡単な説明

第１図は本発明のプログラム表示方法およびプログラム生成方法の一実施例における処理の概略を示すブロック図、第２図(１)はＬＩＳＰプログラムの例示図、第２図(２)は同図(１)のＬＩＳＰプログラムを本発明のプログラム表示方法で表わした例示図、第３図(１)は FORTRAN プログラムの例示図、第３図(２)は同図(１)の FORTRAN プログラムを従来のＰＡＤで表わした例示図、第３図(３)は同図(１)の FORTRAN プログラムを従来のＮＳチャートで表わした例示図、第４図はＰＬ／Ｉプログラムの例示図、第５図(１)はＬＩＳＰプログラムの例示図、第５図(２)は同図(１)のＬＩＳＰプログラムを従来のプログラム表示方法で表わした例示図、第５図(３)は同図(１)のＬＩＳＰプログラムを従来のプログラム表示方法で表わした例示図、第６図(１)はＬＩＳＰプログラムの例示図、第６図(２)は同図(１)のＬＩＳＰプログラムを従来のプログラム表示方法で表わした例示図、第７図(１)は本発明のプログラム表示方法およびプログラム生成方法を実施する装置の斜視図、第８図(１)はＣによるプログラムの例示図、第８図(２)は同図(１)のプログラムを本発明のプログラム表示方法により表示した例示図、第９図(１)はＰＬ／Ｉによるプログラムの例示図、第９図(２)は同図(１)のプログラムを本発明のプログラム表示方法により表示した例示図、第１０図は第６図(１)のＬＩＳＰプログラムを本発明のプログラム表示方法により表示した例示図、第１１図(ａ)(ｂ)は本発明のプログラム表示方法およびプログラム生成方法で用いる変換規則の図表、第１２図は第１図に示す変換プログラムの基本処理のＰＡＤ図、第１３図は第１２図のステップ１０１２２の基本処理のＰＡＤ図、第１４図は第１２図のステップ１０１２１ｂの基本処理のＰＡＤ図、第１５図は第１２図のステップ１０１２１ｅの基本処理のＰＡＤ図、第１６図は第１２図のステップ１０１２１ａの基本処理のＰＡＤ図、第１７図は第１２図のステップ１０１２１ｃの基本処理のＰＡＤ図、第１８図は第１２図のステップ１０１２１ｄの基本処理のＰＡＤ図、第１９図は第１２図のステップ１０１２２の基本処理に含まれる一つの処理のＰＡＤ図、第２０図は第１２図のステップ１０１２２の基本処理に含まれる一つの処理のＰＡＤ図、第２１図は第１２図のステップ１０１２２に含まれる一つの処理の基本処理のＰＡＤ図、第２２図(１)はデータを示すリストの例示図、第２３図(２)は同図(１)のデータを本発明のプログラム表示方

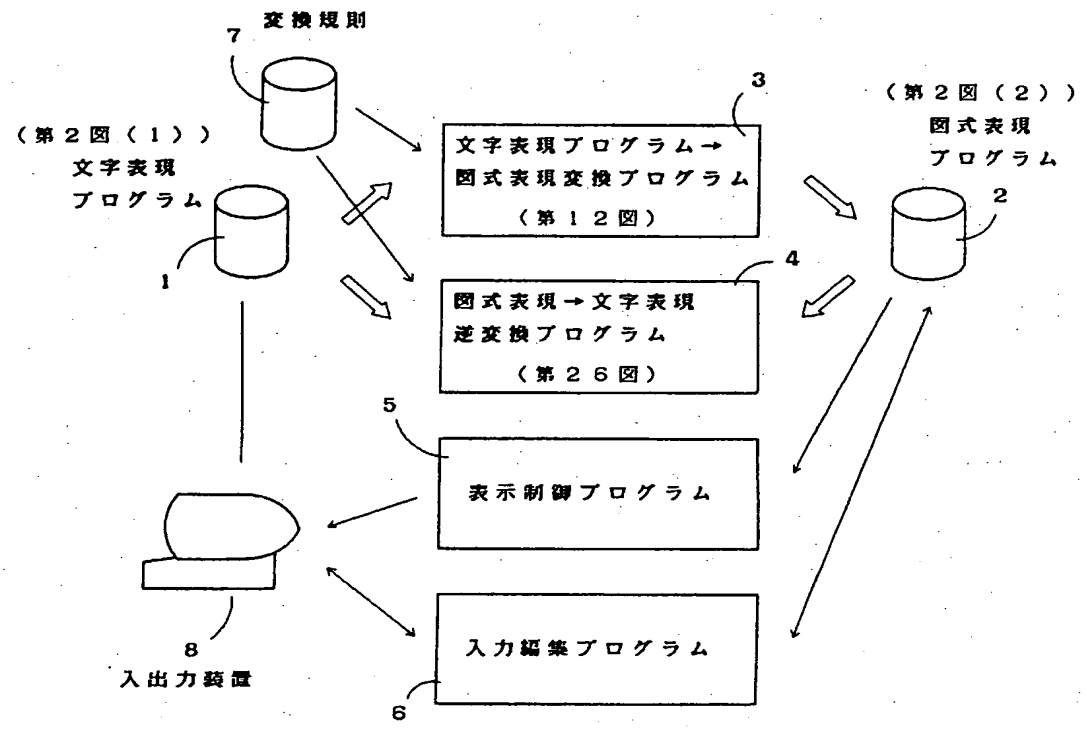
法により表示した例示図、第２３図(１)(２)は第１２図のステップ１０１２２に含まれる一つの処理のＰＡＤ図、第２４図は本発明のプログラム表示方法の他の実施例による図式表示の例示図、第２５図は構造データの例示図、第２６図は第１図に示す逆変換プログラムの基本処理のＰＡＤ図、第２７図(１)は他を参照することを示す図形を表示した状態の例示図、第２７図(２)は参照される内容の図式表現の例示図、第２７図(３)は同図(1)の参照箇所に同図(２)の参照内容を埋め込んで表示した状態の例示図である。

【符号の説明】

１…文字表現プログラム、
２…図式表現されたプログラム、
３…文字表現プログラムから図式表現への変換プログラム、
４…図式表現から文字表現プログラムへの逆変換プログラム、
５…表示制御プログラム、
６…入力編集プログラム、
７…変換規則、
８…入出力装置。

出願人　株式会社　日立製作所
代理人　弁理士　有近　紳志郎

（第１１図（ａ）（ｂ））　　第　１　図



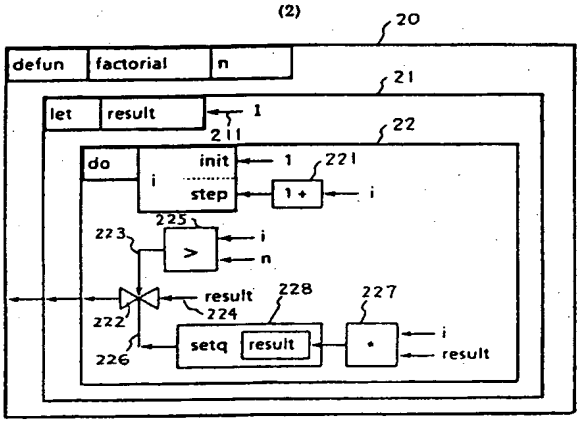
第2図

(1)

```
(defun factorial (n)                          ..... 1
  (let ((result 1)))                          ..... 2
    (do ((i 1 (1+ i)))                        ..... 3
        ((> i n) result)                      ..... 4
      (setq result (* i result)))))           ..... 5
```

行番号

(2)

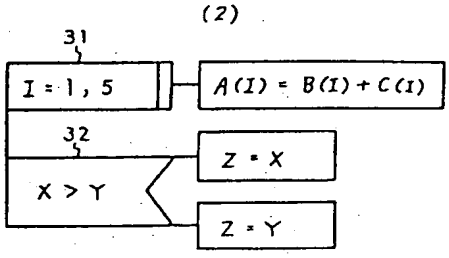


第3図

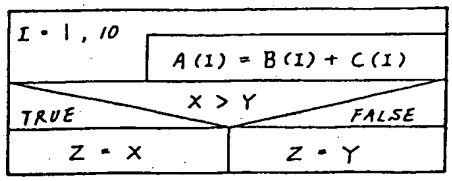
(1)

```
      DO  10  I = 1, 5
        A(I) = B(I) + C(I)
10    CONTINUE
      IF (X.GT.Y) THEN
        Z = X
      ELSE
        Z = Y
```

(2)



(3)



第4図

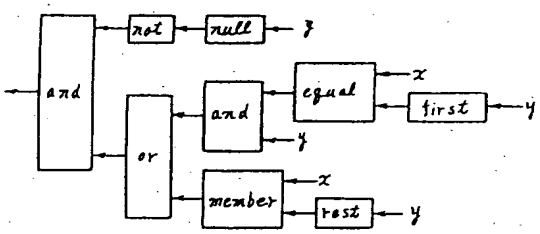
```
                                              行番号
PROCEDURE OPTIONS(MAIN);                      ..... 1
   FUN1 : PROCEDURE (A, B) RETURNS (FIXED BINARY;  ..... 2
      DECLARE (A, B) FIXED BINARY;            ..... 3
      RETURN (A + B);                         ..... 4
   END;                                       ..... 5
   DECLARE C FIXED BINARY;                    ..... 6
   C = FUN1 (5, 3);                           ..... 7
   BEGIN;                                     ..... 8
      DECLARE C FIXED BINARY;                 ..... 9
      C = FUN1 (2, 4);                        .... 10
   END;                                       .... 11
   PUT C;                                     .... 12
END;                                          .... 13
```

第5図

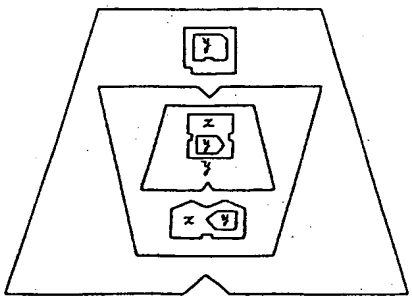
(1)

```
(and (not (null y))
     (or (and (equal x (first y)) y)
         (member x (rest y))))
```

(2)



(3)

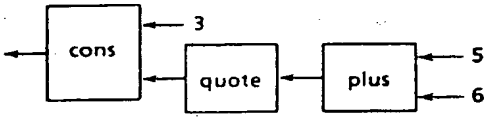


第6図

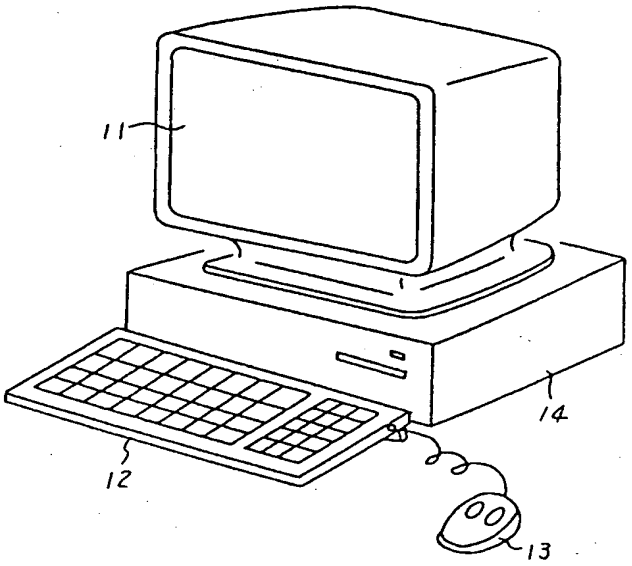
(1)

```
(cons 3 (quote (plus 5 6)))
```

(2)



第 7 図

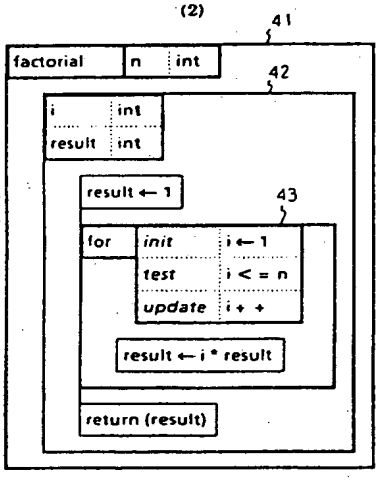


第 8 図

(1)

```
factorial (n)
int n;
{
    int i, result;
    result = 1;
    for (i = 1; i <= n; i++)
        result = i * result;
    return (result);
}
```

(2)

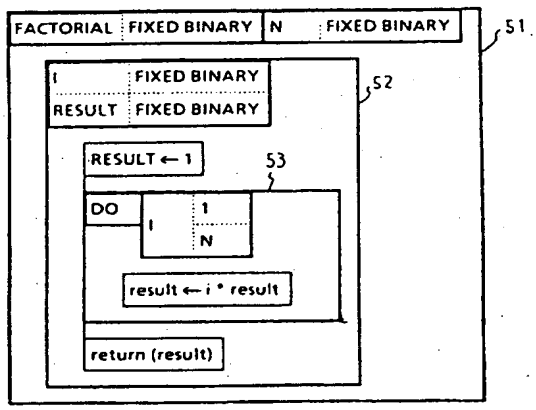


第 9 図

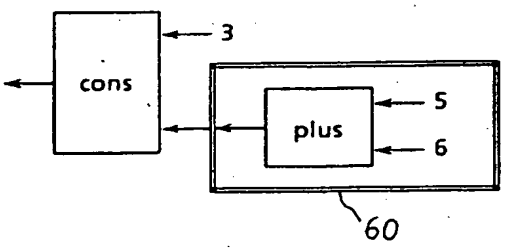
(1)

```
FACTORIAL : PROCEDURE (N) RETURNS (FIXED BINARY);
   DECLARE N FIXED BINARY;
   BEGIN;
      DECLARE (I, RESULT) FIXED BINARY;
      RESULT = 1;
      DO I = 1 TO N;
         RESULT = I * RESULT;
      END;
      RETURN (RESULT);
   END;
END;
```

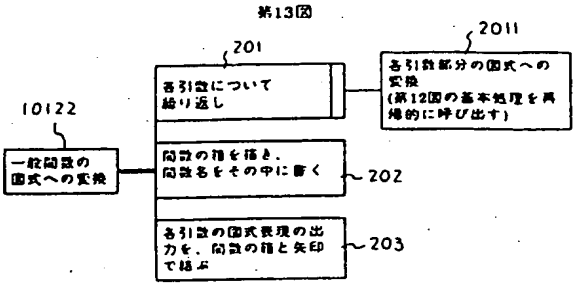
(2)



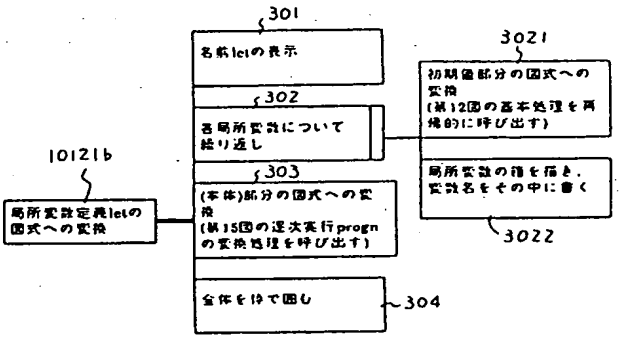
第10図



第13図



第14図



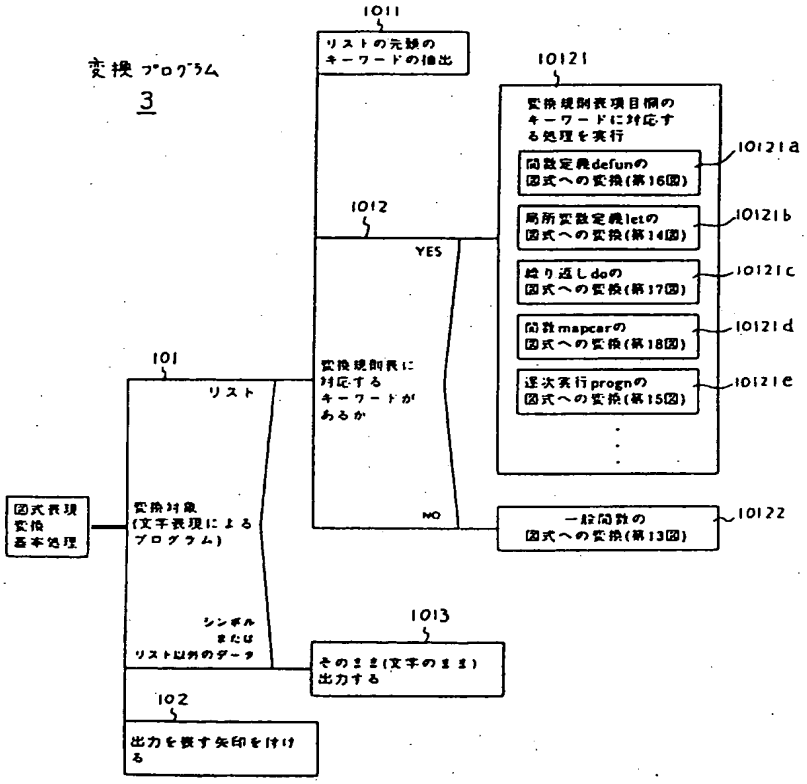
第11図(a)

No	項目	文字表現	図式表現	該当図
1	一般の関数	((関数名) (引数1) (引数2) ……)	(戻り値) (関数名) (引数1) (引数2)	第13図
2	関数定義 (defun)	(defun (関数名) ((引数1) (引数2) …) (本体))	defun (関数名) (引数1) (引数2) (本体)	第16図
3	局所変数定義 (let)	(let (((変数1) (初期値1)) ((変数2) (初期値2)) …) (本体))	let (変数1) (初期値1) (変数2) (初期値2) (本体)	第14図
4	繰り返し (do)	(do (((変数1) (初期値1) (ステップ値1)) ((変数2) (初期値2) (ステップ値2)) …) ((終了テスト) (結果)) (本体))	do (変数1) init (初期値1) step (ステップ値1) (変数2) init (初期値2) step (ステップ値2) (終了テスト) (結果) (本体)	第17図
5	map関数 (mapcar)	(mapcar (関数) (リスト))	mapcar (関数) (リスト)	第18図

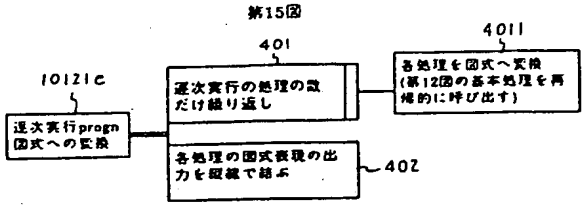
第11図(b)

No	項目	文字表現	図式表現	変換処理
6	逐次実行 (progn)	(progn (処理1) … (処理n))	(処理1) (処理n)	第15図
7	条件分岐 (if)	(if (条件) (処理1) (処理2))	(条件) (処理1) (処理2)	第19図
8	代入 (setq)	(setq (変数) (処理))	setq (変数) (処理)	第20図
9	引用 (quote)	(quote (対象)) または '(対象) [注] [']は引用符を示す。	(対象)	第21図
10	バック コート	例：`(list ,(+ x y) ,(* z w)) [注] [`]は逆引用符を示す。	x y z w	第23図
11	シンボル	(シンボル名)	(シンボル名)	
12	データ	例1) 数値 123 例2) 文字列 "abc"	123 "abc"	

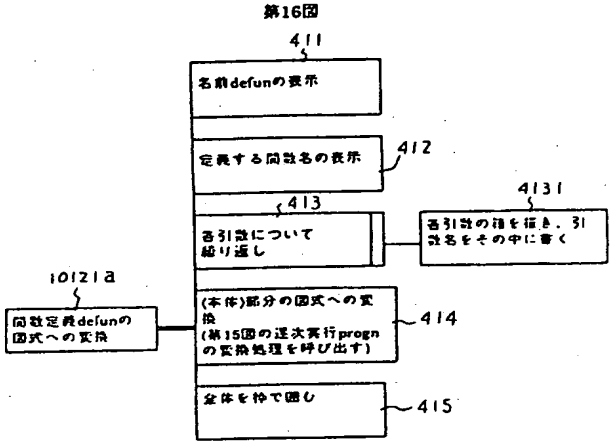
第12図



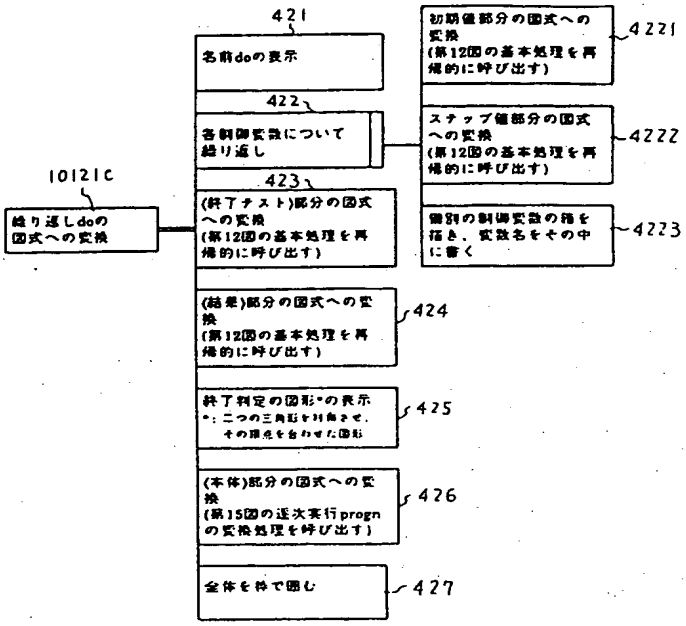
第15図



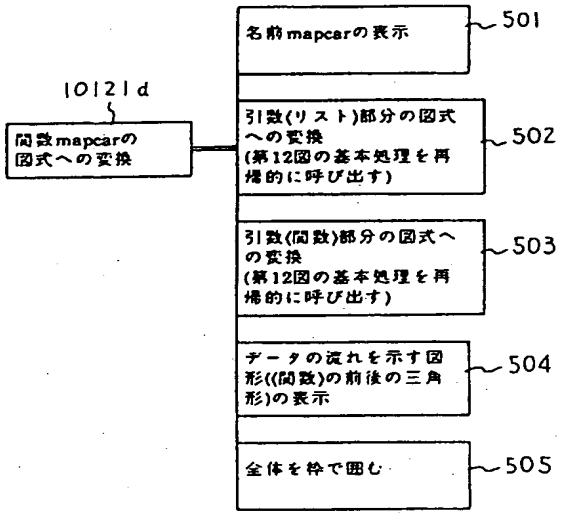
第16図



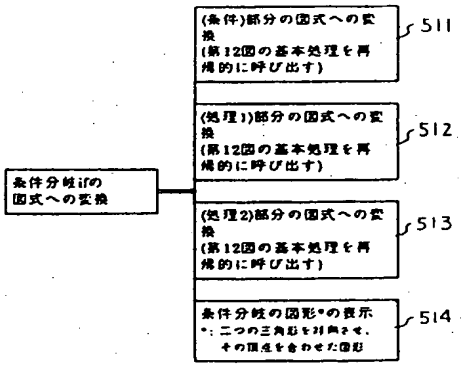
第17図



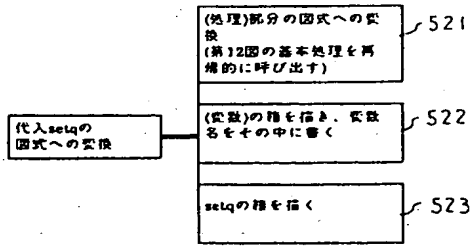
第18図



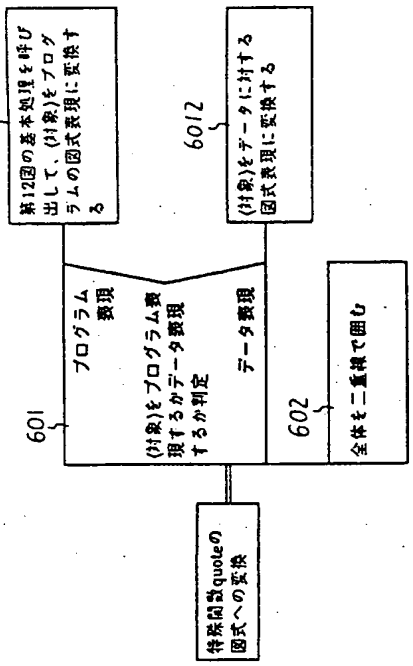
第19図



第20図



第21図

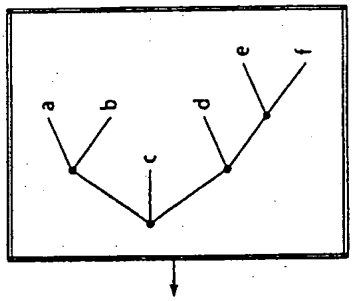


第22図

(1)

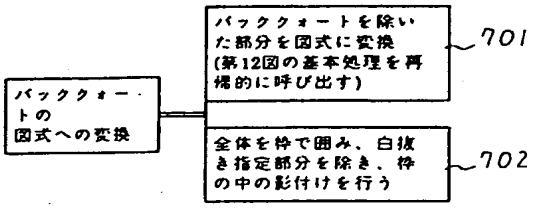
(quote ((a b) c (d (e f))))

(2)

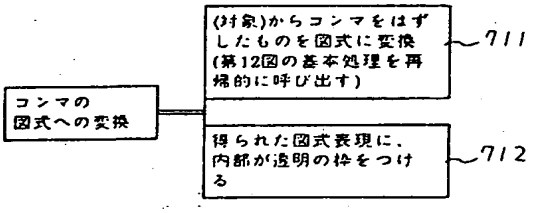


第23図

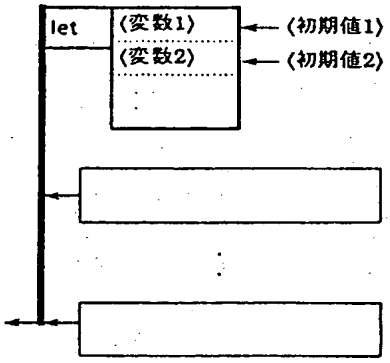
(1)



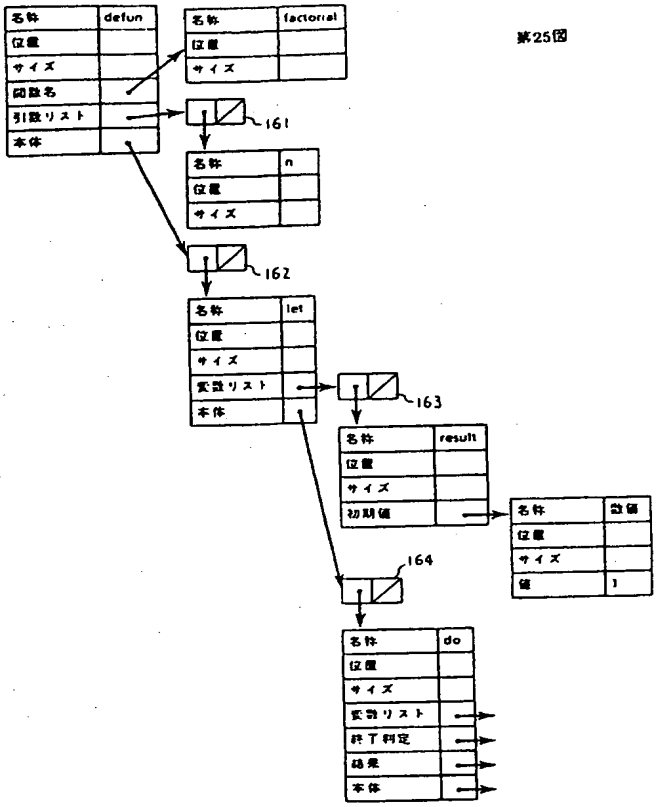
(2)



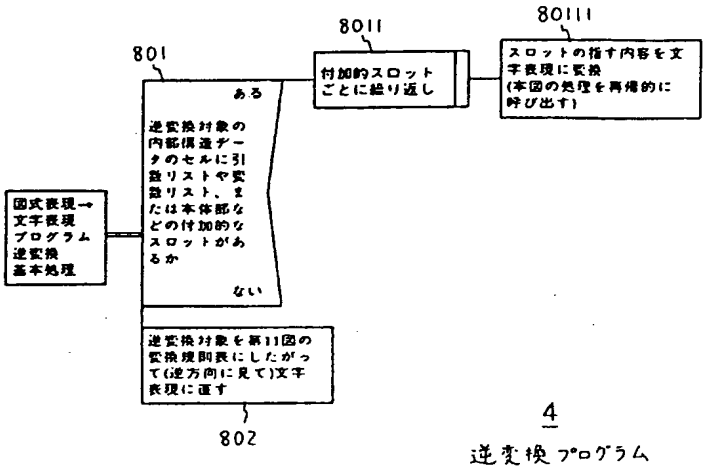
第24図



第25図



第26図

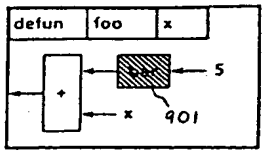


4

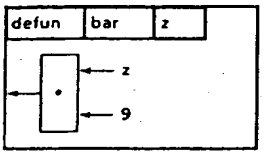
逆変換プログラム

第27図

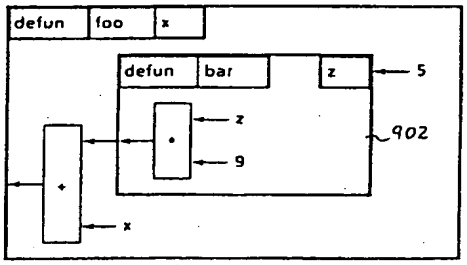
(1)



(2)



(3)



手続補正書 (方式)

平成3年10月11日

特許庁長官殿

1. 事件の表示
   平成2年特許出願第281269号
2. 発明の名称
   プログラム表示方法および装置ならびに
   プログラム生成方法および装置
3. 補正をする者
   事件との関係　特許出願人
   住所　東京都千代田区神田駿河台四丁目6番地
   名称　(510) 株式会社 日立製作所
   代表者　金井 務
4. 代理人
   住所　169 東京都新宿区高田馬場1－18－26
   ロイヤルハイネス 103号
   TEL 03-3209-6777 FAX 03-3208-0644
   氏名　弁理士(9551) 有近 紳志郎
5. 補正命令の日付（発送日）　平成3年10月8日
6. 補正の対象
   明細書の「図面の簡単な説明の欄」
7. 補正の内容
   明細書の第50ページの下から2行目の「第23図（2）」を「第22図（2）」に訂正する。